# LISTA ENLAZADA SIMPLE

Una lista enlazada es una colección ordenada de elementos de datos. Un elemento de datos se puede representar como un nodo en una lista enlazada. Cada nodo consta de dos partes: datos y puntero al siguiente nodo.

A diferencia de los arreglos, los elementos de datos no se almacenan en ubicaciones contiguas. Los elementos o nodos de datos están vinculados mediante punteros, por lo que se denomina lista enlazada.

**Una lista enlazada tiene las siguientes propiedades:**

* Los nodos sucesivos están conectados por punteros.
* El último nodo apunta a null.
* En head Se mantiene el puntero que apunta al primer nodo de la lista.
* Una lista vinculada puede crecer y reducirse durante la ejecución del programa.
* Asigna memoria a medida que crece la lista. A diferencia de los arreglos, que tienen un tamaño fijo. Por lo tanto, el límite superior del número de elementos debe conocerse de antemano. Generalmente, la memoria asignada es igual al límite superior independientemente del uso. Ésta es una de las ventajas clave de usar una lista enlazada sobre un arreglo.

A diferencia de un arreglo, que almacena datos de forma contigua en la memoria, una lista enlazada puede insertar o eliminar fácilmente nodos de la lista sin reorganizar toda la estructura de datos.

**Algunos inconvenientes:**

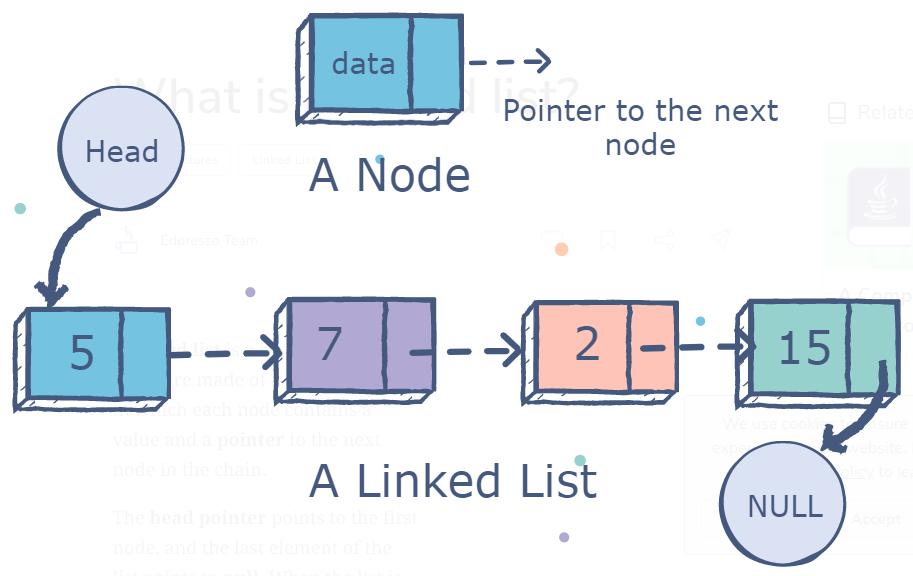
No se permite el acceso aleatorio de elementos de datos. Se debe acceder a los nodos de forma secuencial comenzando desde el primero. Por lo tanto, la operación de búsqueda es lenta en una lista enlazada.

Utiliza más memoria que los arreglos debido al almacenamiento que utilizan sus punteros.

Tipos de listas enlazadas

Hay algunos tipos diferentes de listas enlazadas. Pero los más populares son: simple, doble y circular.

Una lista enlazada simple es una colección de nodos en la que cada nodo tiene 2 partes: datos y un puntero al siguiente nodo. La lista termina con un nodo apuntando null.



## clase Nodo.

Definamos una clase de Nodo y veamos cómo esta clase de Nodo, junto con la clase de listaEnlazada, ayudará a construir una lista lineal de nodos.

//clase para crear los nodos

class Nodo{

    constructor(dato, next = null){

        this.dato = dato;

        this.next = next;

    }

}

En el código anterior, se define una clase de Nodo. Cuando se forma una instancia de la clase Nodo, se llamará a la función constructora para inicializar el objeto con dos propiedades, dato y un puntero nombrado next. El puntero next se inicializa con un valor predeterminado de null, en caso de que no se pase ningún valor como argumento.

## clase ListaEnlazada

a continuación, una clase de ListaEnlazada que mantiene el head puntero de la lista.

//clase para crear la lista

class ListaEnlazada{

    constructor(){

        this.head = null;

        this.size = 0;

    }

Se define una clase ListaEnlazada. Cuando se forma una instancia de la clase ListaEnlazada, se llamará a la función constructora para inicializar el objeto con dos propiedades head y size.

Al head puntero se le asigna un valor de null porque cuando se crea inicialmente un objeto de lista enlazada, no contiene ningún nodo. Es cuando agregamos nuestro primer nodo a la lista enlazada cuando lo asignaremos al head puntero, la propiedad de size está asignada en cero porque no hay ningún elemento, al momento de agregar elementos esta se incrementará.

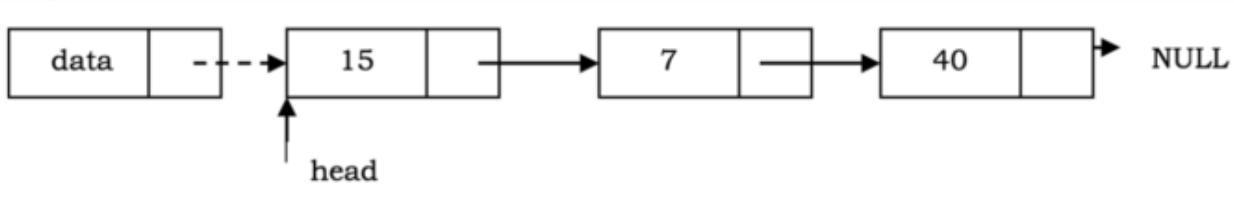
## Insertar elementos en una lista enlazada simple.

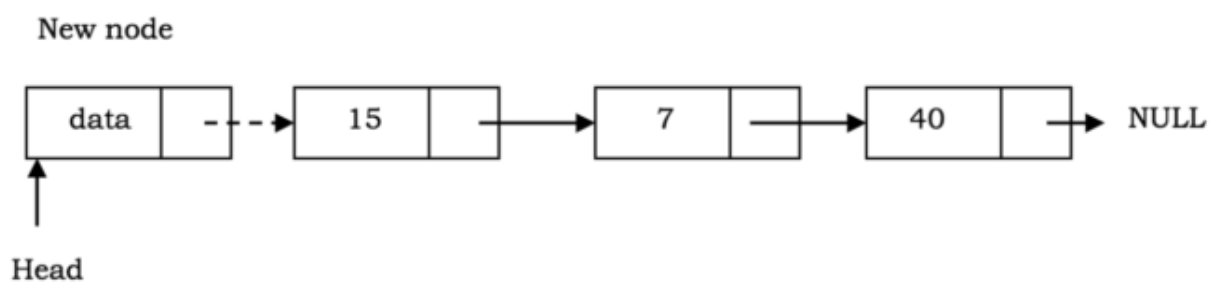
Una operación de inserción agregara un nodo en la lista. Puede haber tres tipos de inserción.

* Insertar un nuevo nodo antes de head (al principio de la lista, **insertBefore**).
* Insertar un nuevo nodo después de la cola (es decir, al final de la lista, **insertAfter**).
* Insertar un nuevo nodo en el medio de la lista (en una posición aleatoria determinada, **insertAt**).

### Insertar un nodo al principio de la lista enlazada simple (insertBefore).

se agrega un nuevo nodo antes del nodo principal actual. Para realizar esta operación, primero crearemos un nodo. El nodo recién creado tendrá dos propiedades definidas en la función constructora de la clase Nodo, dato y next.





insertBefore(dato){

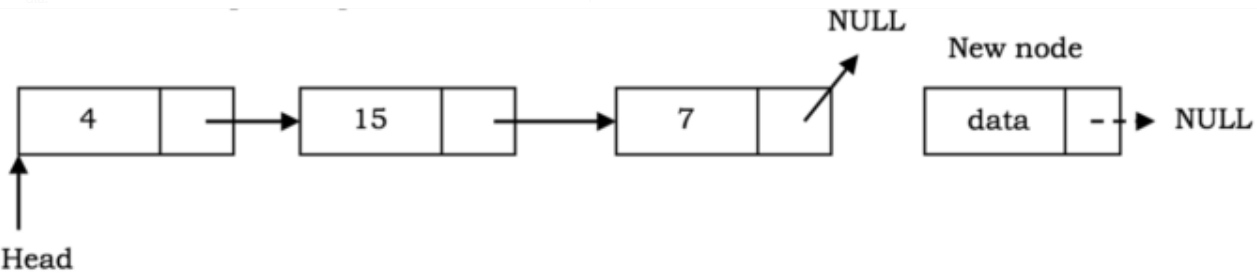
        this.head = new Nodo(dato, this.head);

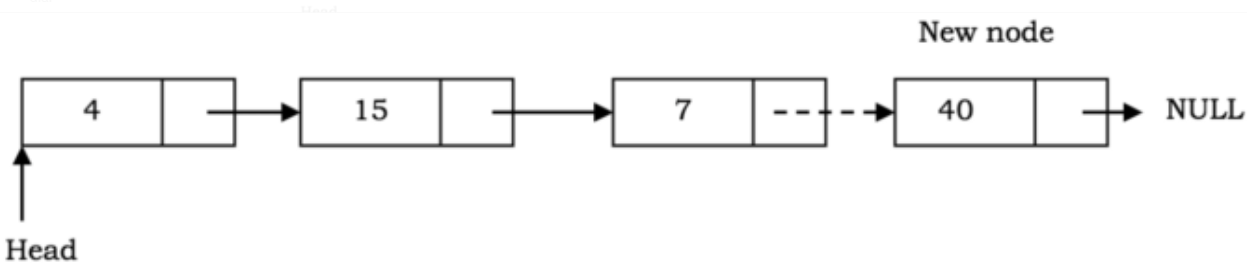
    }

### **Insertar un nodo al final de la lista enlazada simple (insertAfter)**.

se agrega un nuevo nodo al final de la lista. Para implementar esta operación, tendremos que recorrer la lista para encontrar el nodo de cola y modificar el siguiente puntero de la cola para que apunte al nodo recién creado en lugar de null.

Inicialmente, la lista está vacía y los head puntos son nulos.





    insertAfter(dato){

        let newNode = new Nodo(dato);

        if(! this.head){

            this.head = newNode;

            return this.head;

        }

        else{

            let tail = this.head;

            while(tail.next !==null){

                tail = tail.next;

            }

            tail.next = newNode;

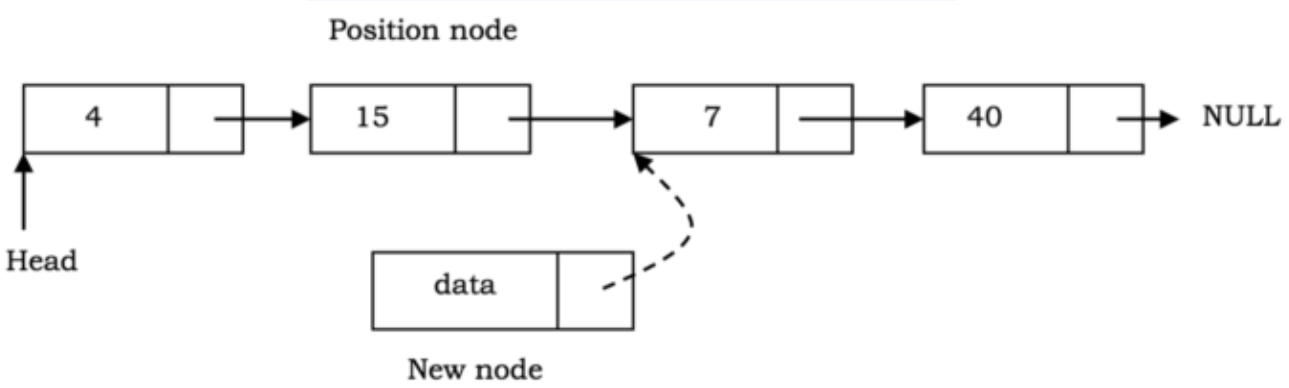
            return this.head;

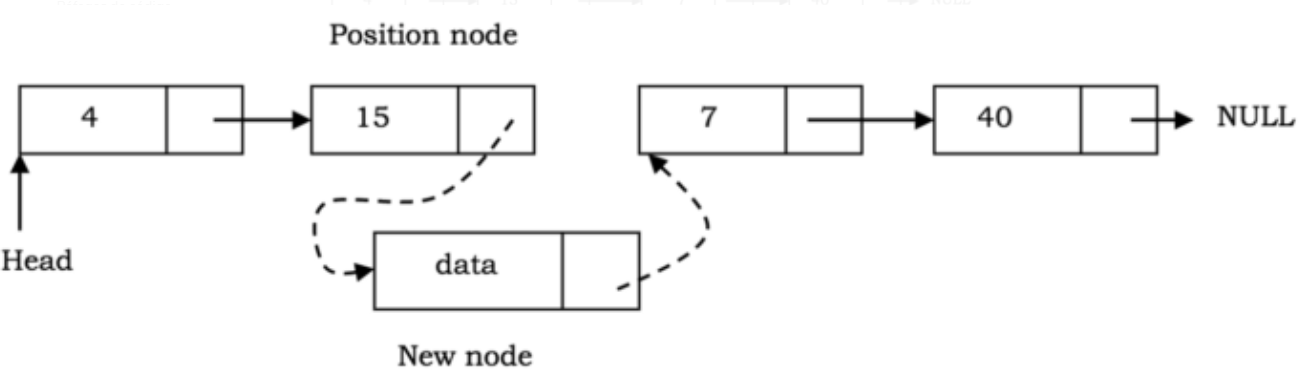
        }

    }

### Insertar un nodo en una posición especifica dada en una lista enlazada simple (insertAt).

Para implementar esta operación tendremos que recorrer la lista hasta llegar al nodo de posición deseado. Luego asignaremos a la propiedad next puntero de newNode el siguiente nodo del nodo de la posición dada. El puntero next del nodo de la posición dada se puede actualizar para que apunte al nuevo nodo.



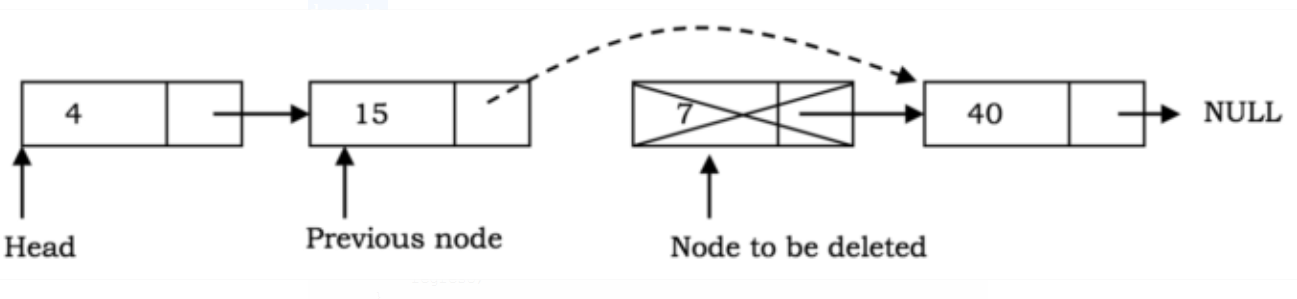


## Eliminar elemento en una lista enlazada simple.

Puede haber 5 casos para la operación de eliminación.

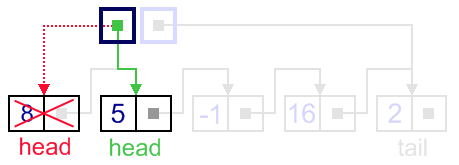
* Remueve el nodo con un valor especifico (**removeData**).
* Remueve el primer nodo (**removeFirst**).
* Remueve el ultimo nodo (**removeLast**).
* Remueve un nodo en específico (**removeFrom**).
* Remueve toda la lista (**removeList**).

### Remover un nodo con un valor especifico (removeData).



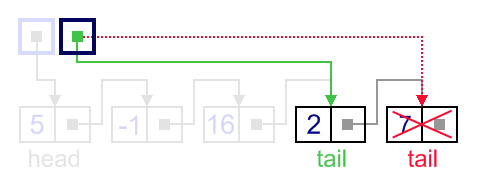
### Remover el primer nodo de una lista enlazada simple (removeFirst).

Se remueve el nodo **head** y **head.next** pasa a ser head ahora.



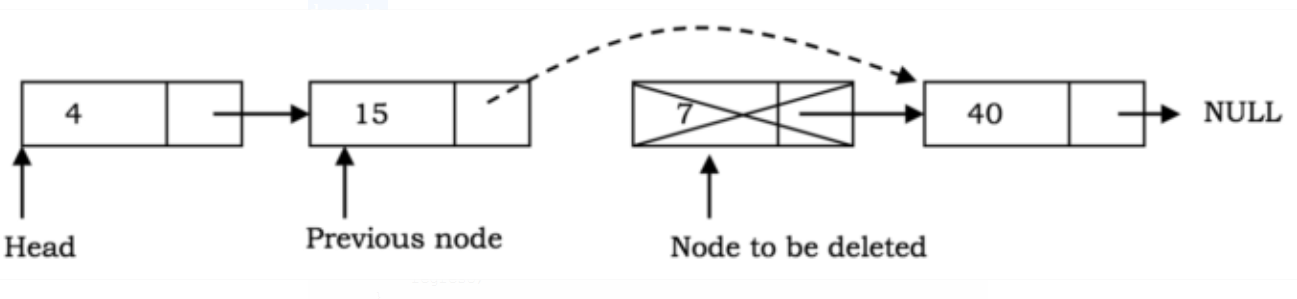
### Remover el ultimo nodo de una lista enlazada simple (removeLast).

Se remueve el nodo que se encuentra al final de la lista.



### Remover un nodo especifico (removeFrom).

Primero tendremos que recorrer la lista para encontrar el nodo deseado que se eliminará y, al mismo tiempo, mantener un puntero adicional para apuntar al nodo anterior del nodo deseado.



### Remueve toda la lista (removeList).

Eliminemos la lista completa vinculada. Esto se puede hacer con una sola línea de código.

    removeList(){

        this.head = null;

    }

## Métodos auxiliares de la lista enlazada

### isEmpty

El método retorna un **true** si la lista esta vacía y un **false** si no lo está.

    isEmpty(){

        if(this.size === 0){

            return true;

        }

        else{

            return false;

        }

    }

### getSize

El método retorna el tamaño de la lista.

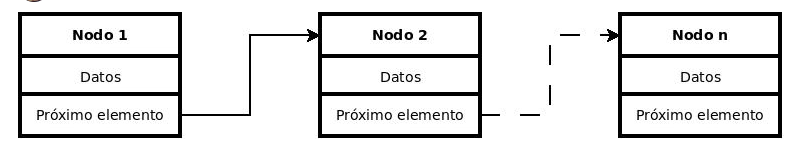
    getSize(){

        return this.size;

    }

### print

El método muestra los valores de nuestros elementos. Esto lo realiza recorriendo la lista con un **while** que se ejecuta mientras la variable **current** tenga contenido, muestra en consola el valor de la propiedad **current.dato** y después la variable **current** apuntara a **current.next**.



consola

Datos

Datos

Datos

        let current = this.head;

        while(current){

            console.log(current.dato);

            current = current.next;

        }

    }